

UNIVERSITÀ DI PISA
DIPARTIMENTO DI INFORMATICA

TECHNICAL REPORT

Assigning and sequencing storage locations under a two level storage policy: optimization model and matheuristic approach

Giacomo Lanza

Mauro Passacantando

Maria Grazia Scutellà

March 18, 2021

LICENSE: Creative Commons: Attribution-Noncommercial - No Derivative Works

ADDRESS: Largo B. Pontecorvo 3, 56127 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

Assigning and sequencing storage locations under a two level storage policy: optimization model and matheuristic approach

Giacomo Lanza Mauro Passacantando Maria Grazia Scutellà

March 18, 2021

Abstract

We deal with a storage location problem in a warehouse where items of different product types are released by the production area and need to be stored. Capacitated storage locations have to be assigned to each product type to store the corresponding items. Items of different product types cannot share the same storage location, i.e., each storage location can be assigned to at most one product type. In addition, a suitable sequencing of the assigned storage locations must be devised for each product type. Each sequence will provide both the order with which the storage locations will be filled up during the storing operations, and also the order of visit of the storage locations during the successive order picking phase. A motivation is that, separately per product type, an order picking based on the time of permanence of the items in the warehouse has to be pursued. Moreover, the chosen sequencing influences the availability of additional storage on the top of the assigned storage locations. In fact, for each product type, an additional extra storage can be made available on top of pairs of consecutive storage locations in the sequence, which depends on the two storage locations at the ground level. The goal is to maximize the storage capacity still available after the assignment of the storage locations. After proving the NP-Hardness of the considered problem, we model it in terms of a multicommodity flow problem with additional constraints on an auxiliary graph, and we propose a Mixed-Integer Linear Programming model for its mathematical formulation. A matheuristic approach, based on the sequential resolution of multicommodity flow subproblems, is then presented. The proposed methodology is applied to a case study related to a large warehouse with a high stock rotation index in tissue logistics, which motivated our study. Computational results on a wide test bed related to such a real application context show the efficiency and efficacy of the proposed approach.

1 Introduction

Warehouses are an essential component of any supply chain. Their operation systems are configured through the following basic processes: reception and dispatch, order picking, storage (Gu et al., 2007; Rouwenhorst et al., 2000). Reception and dispatch are the interface of a warehouse for incoming and outgoing material flow, and concern the organization of all the operations required to manage entering and exiting *Stock Keeping Units* (SKUs). Order picking is generally recognized as the most expensive warehouse operation, as it tends to be either very labor intensive or very

capital intensive. It requires the organization of the orders to be picked up and of the resources needed for picking. Storage is concerned with the organization of goods held in the warehouse in order to achieve high space utilization and facilitate efficient material handling. The broad organization of goods in a warehouse is normally a strategic/tactical decision made on management (such as a dedicated storage area for a specific product type) or material handling considerations (such as a forward area for fast picking). This process results into a long-term *storage assignment policy*, which further defines the internal configuration of the warehouse (such as dimension of specific storage areas and aisle configurations) and that fixes the rules to follow when stocking of products is needed.

There are various storage assignment policies described in the literature (Gu et al., 2010; Roodbergen and Vis, 2009). The most representatives are the random, the dedicated, and the class-based allocation policies (see the pioneering studies of Hausman et al., 1976; Ashayeri and Gelders, 1985). The random policy involves the random assignment of SKUs to any available and eligible location within the storage area, each location having an equal probability of being selected. In the dedicated assignment policy, the warehouse is divided into a number of zones, each of those dedicated to one product type. Replenishment of a product always occurs at the corresponding dedicated zone. In the class-based policy, products are classified into a number of classes based on their properties (such as demand rate, order frequency, dimension, or product correlations), each product class having a reserved zone within the warehouse. Accordingly, an incoming SKU is stored at an arbitrary available location within its reserved zone.

Once a long-term storage assignment policy has been defined, it is necessary to determine the exact physical location of the SKUs in the warehouse, possibly subject to additional rules depending on the specific application context. The associated problem is generally known as the *Storage Location Assignment Problem* (SLAP). Specifically, given information on the availability, physical dimension and position of storage locations, and given information on the set of items to be stored (such as product type and physical dimension), SLAP aims at determining the physical locations where items have to be allocated in the warehouse by optimizing criteria such as material handling cost or storage space utilization, while respecting the storage assignment policy chosen at a strategic/tactical level and specific assignment rules, if present. SLAP is an operational decision problem, which has a strong influence on other decisions within the warehouse, such as order picking and routing aspects.

In this paper, we address a problem which combines storage location assignment with sequencing decisions about the assigned storage locations, and which originates from a real-world application context in tissue logistics. Specifically, given a set of different product types, each with its own storage demand expressed in number of SKUs to store in a given time horizon, a set of storage locations has to be assigned to each product type for the corresponding storing operations. Each storage location has a capacity which depends on the product type, i.e., a maximum number of SKUs can be stored for that product type, and it can be assigned to at most one product type, i.e., different types of products cannot share the same storage location. In addition, a suitable sequencing of the assigned storage locations must be devised for each product type, i.e., it has to be decided the ordering with which the storage locations will be filled up during the storing operations. A motivation is that an order picking based on the time of permanence of the items in the warehouse has to be pursued. More precisely, a FIFO (First-In First-Out) picking criterion

among storage locations is required per product type. That is, separately per product type, SKUs stored in a certain storage location cannot be retrieved if SKUs in previously replenished storage locations have still to be picked up. Notice that the FIFO picking criterion is not required inside storage locations: in that case, the order of picking will depend on the specificity of the considered storage locations. The sequencing established for the assigned storage locations will thus allow to easily implement the FIFO policy in the successive order picking steps. Moreover, for each product type, the selected sequencing also determines the availability of additional extra storage for that product type. Specifically, an additional amount of storage can be made available on the top of pairs of consecutive storage locations along the sequence, provided that they are fully replenished and physically contiguous. The amount of the available storage on the top does depend on the two storage locations at the ground level and on the product type to be stored. Additional soft constraints are present, as better clarified next, with the goal of maximizing the storage capacity which remains available after the assignment of the storage locations.

After proving its NP-Hardness, we formulate the considered problem in terms of constrained multicommodity flows on an auxiliary graph, and we propose a Mixed-Integer Linear Programming (MILP) model, together with some model enhancements, based on the suggested multicommodity flow formulation. Since the problem can be very hard to address computationally, a matheuristic approach is designed starting from the multicommodity flow based MILP model. A case study is then presented, which is related to the tissue logistics sector and which motivated our research in this topic. The involved warehouse is larger than 10,000 m² and is characterized by a high product rotation index (specifically, more than 1,000 pallets are moved per day). Its modernization is the goal of a big research project funded by Regione Toscana, in Italy, and it includes the resolution of the considered combined assignment-sequencing storage location problem via Operations Research techniques. Computational experiments on real data provided by the company show the efficiency and the efficacy of the proposed approach.

The paper is organized as follows. Section 2 reviews the main results from the literature. Specifically, the main results in the area of SLAPs are presented in Section 2.1. Since storage locations are stacks in the presented case study, storage systems based on stacks are reviewed in Section 2.2, as well as near systems such as the deep-lane ones. Multi-level storage assignment features are also discussed. Moreover, works taking into account picking aspects when assigning storage locations are presented in Section 2.3, positioning then our contribution with respect to the literature in Section 2.4. Section 3 describes the problem addressed in this paper in more detail, and provides the corresponding NP-Hardness result. Section 4 presents the multicommodity flow based MILP formulation for the considered problem. The matheuristic approach built to tackle the problem is described in Section 5. Section 6 presents the case study and describes the experimental plan, by reporting the results of the computational experiments we performed. Finally, Section 7 concludes the paper and identifies some future directions of research.

2 Literature review

Storage assignment problems, also referred to as loading problems in a broader perspective (Lehnfeld and Knust, 2014), deal with the storage of incoming items. Each item reaching the storage area, which can be a warehouse, but at a more general level could be a yard, the bunt of a con-

tainer ship or even a tram/bus depot, has to be assigned to a feasible location and stored until it is required to be retrieved. A storage assignment plan decides on the exact storage position of each SKU in the storage system (Lehnfeld and Knust, 2014). As indicated before, such decisions are made by considering some long-term storage assignment policy (random, dedicated and class-based policy, previously mentioned, are the most popular), that broadly prescribes the rules to follow when SKU stocking is needed, and which strictly depends on the considered application. An overview of container assignment policies in terminals can be found in Dekker et al. (2007) and Stahlbock and Voß (2008). Focusing on traditional (aisle-based) warehouses instead, an overview of warehouse management policies can be found in Rouwenhorst et al. (2000); De Koster et al. (2007) and Gu et al. (2007). Operationally, besides the chosen storage assignment policy, storage decisions are made by considering criteria related to material handling costs, stocking/retrieving efforts, time for order preparations, resource utilization for stocking/picking operations, warehouse space utilization or even energy consumption.

Since storage assignment problems originate from a wide range of different applications, a lot of scientific literature exists dealing with problems motivated from practice. These problems are often described with different terms and notions depending on the characteristics of the storage system and on the specific context, as overviewed next.

2.1 Storage Location Assignment

When items have to be stored on racks or shelves accessible from the side, then the problem is commonly referred to as the *Storage Location Assignment Problem*, or SLAP, as previously introduced (Rouwenhorst et al., 2000; De Koster et al., 2007; Gu et al., 2007). The SLAP literature is quite diversified because of the variety of peculiar storage assignment policies (Li et al., 2016; Pang and Chan, 2017), storage system customization (Bortolini et al., 2015), product management rules (Zhang et al., 2017), internal layouts of warehouses (Ramtin and Pazour, 2015; Foroughi et al., 2020) and optimization criteria (Meneghetti and Monti, 2014; Yang et al., 2015; Battini et al., 2016; Ene et al., 2016; Larco et al., 2017) requested in real situations. We mention here Quintanilla et al. (2015), who consider a real problem raised by a Spanish company, where pallets of different weights need to be stocked in a chaotic warehouse. The warehouse has a rectangular layout with a certain number of racks on both sides of a set of parallel storage aisles. Racks may accommodate SKUs on a double-level (one on top of the other) and on double-depth (one behind another). However, heaviest pallets should be stored below a maximum level and the most fragile ones cannot have another pallet on top of them. The crucial point to manage for the company is the remaining available storage capacity after the allocation of the SKUs. The authors propose a MILP model with constraints on positions and level restrictions, whose objective is maximizing the space available for future assignments after the current SLAP is solved. Several heuristic algorithms and a local search procedure are presented.

2.2 Stacking, Deep-Lane and Multi-level Storage Assignment

Some storage systems consist of multiple stacks where items are stored on top of (e.g., a pile of containers) or back-to-back to (e.g., a queue of pallets) each other. These systems are typically considered when dealing with the storage of high volumes of SKUs having large inventory and high turnover (Accorsi et al., 2017). When items are piled one on top of another on vertical

stacks, and cranes are employed to move the SKUs and can only access the topmost ones, as in case of containers or steel slabs in yards, the problem is usually referred to as a *Stacking problem* (Lehnfeld and Knust, 2014). On the other hand, when items are stored on horizontal stacks on the ground, such as SKUs stored back-to-back of each other on queue accessible only frontally, then the problem is known as a *Deep-Lane Storage Assignment problem* (Boysen et al., 2018). The two problems are extremely similar since the horizontal height of stacks in the first problem corresponds to the vertical depth of deep-lanes in the second one.

Considering the first kind of problems, container stacking problems have been widely studied and reviews may be found in Vis and De Koster (2003); Dekker et al. (2007); Stahlbock and Voß (2008) and Carlo et al. (2014). They may raise in storage yards, where containers are stored temporarily after they are discharged from vessels or before they are loaded onto vessels, or in vessels themselves, where containers are stowed and additional ship stability constraints are thus imposed.

On the other hand, focusing on deep-lane storage systems, in different applications SKUs may be not only stored back-to-back of each other, but also one on top of each other. The so-called stackability in a deep-lane context, defines the number of levels a deep-lane may have to store SKUs by still maintaining the storage safety (Accorsi et al., 2017). In particular, Zaerpour et al. (2015) consider a multi-level deep-lane storage system where unit loads are stored on deep-racks following a shared storage policy, i.e., different product types share the same lane. Moreover, the assignment of SKUs to a deep-lane belonging to a certain level may be done no matter if the lower levels are already occupied.

The multi-level feature is particularly addressed in the steel industry, where coils can usually be stored one on top of the others up to two levels for stability and safety. At the ground level a coil may be stored at any available position in the storage area, while at the upper level a coil can only be stored on top of two adjacent ground-level coils in the same row. Thus, coils are triangularly related, while normally SKUs on stacks are vertically related (Tang et al., 2012). In Zäpfel and Wasner (2006) the storage assignment of steel coils considering such a storage policy is accounted for. Coils are of different types and the only restriction to exploit the triangular storage policy is that a coil can be positioned in an upper storage location if the location is empty and both locations underneath contain coils, no matter of the product type. A unique overhead traveling crane moves coils to perform storage operations simultaneously with retrieval and reshuffling operations. The problem is compared to a job shop scheduling problem and formulated as a nonlinear integer programming model aiming at minimizing the completion time for the last order during the planning period. A local search based heuristic is proposed and tested through computation. Tang et al. (2015) consider the storage assignment of coils onto the stowage of a ship. The authors formulate the problem as a MILP aiming at minimizing a combination of ship instability throughout the entire voyage, the shuffles needed for unloading at the destination ports, and the dispersion of coils in the stowage destined to the same port. A construction heuristic coupled with a tabu search algorithm is developed, and several valid inequalities are proposed to help reducing the solution time. Other applications where this special stacking structure is considered are Tang et al. (2012, 2014); Xie et al. (2014) and Maschietto et al. (2017), where the focus however is the reassignment of locations for those coils blocking other ones targeted to retrieve.

2.3 Storage locations assignment addressing retrieval order aspects

Frequently, when stacks are filled up with not interchangeable SKUs, some specific retrieval orders, such as the Last-in First-Out (LIFO) or the FIFO one, are requested to be followed when picking SKUs to fulfill the orders requested (Gu et al., 2007). In particular, the FIFO policy is often taken into account, especially for those products with a peculiar product life (like fresh or frozen food). Even though this is a common practice in many real situations, the required retrieval order is normally addressed only when picker routing is planned, and often only approximations are used (Pang and Chan, 2017; Accorsi et al., 2017). Instead, no specific actions are usually taken when storage locations are assigned to SKUs to ease guiding retrieval operations later on, when orders are to be satisfied.

To the best of the author’s knowledge, very few contributions deal with this aspect. Some of them have already been reviewed before, focusing however on the kind of storage system used rather than on picking order considerations. These approaches address the problem of storing SKUs in such a way as not to block other retrieving SKUs, or at least to minimize blocking situations, where a SKU is said to be blocked if one or more SKUs with later retrieval times are stocked above/in front of it. To retrieve the current target SKU thus, other ones need to be removed from the current stack and located into other positions through reshuffling and premashalling operations (Carlo et al., 2014; Lehnfeld and Knust, 2014).

Revillot-Narváez et al. (2020) consider a compact storage system composed of multi-level deep-racks accessible from the front, where SKUs of different product types share the same lane. When pallets are stored, an ascending ranking is reported on their tags, and their retrieval has to be performed by strictly following this FIFO order. The authors develop two ILP models (with and without premarshalling) to assign locations to SKUs aiming at minimizing the number of reshuffles over the planning horizon. Two greedy-randomised heuristic approaches are proposed to solve large real-size instances, which are tested on realistic data from a frozen food distribution centre in Chile.

Slightly different are the approaches of Zaerpour et al. (2015); Boysen et al. (2018) and Boywits and Boysen (2018), that consider a specific procedure to store SKUs grouped in orders into a set of deep lanes. In particular, to enable a smooth retrieval process, orders are stored based on the arrival time (or time windows) of the truck in which they will be loaded. No two pallets destined to different trucks with overlapping retrieval intervals are assigned to the same lane. Zaerpour et al. (2015) consider multi-level deep-lane racks with frontal and posterior access, where a FIFO retrieval policy is followed. Earlier orders are positioned first, while later orders as last in the system. The authors propose a MILP formulation whose objective is to minimize the total retrieval time, and a three step constructive heuristic to solve real instances. Boysen et al. (2018) consider a problem in distribution centers handling fresh produce. A deep-lane refrigerated storage system is considered, all lanes having identical capacity and being initially empty. Pallets of food assembled during the day according to the demands of supermarkets have to be intermediately stored until the next morning, when trucks servicing the supermarkets have to be loaded. A storage assignment is sought such that blockings are avoided and the minimum number of lanes is utilized. The authors investigate the most usual case in which deep-lanes allow only a front access compared to a novel system in which access is allowed from both sides. In the first case, a LIFO policy for retrieval needs to be considered: later orders are positioned on the back while earlier

ones at the front of the system. In the second case, a FIFO policy for retrieval is applied. The authors propose two MILP models and provide a simple yet effective solution procedure based on a problem decomposition. A robust against unpunctual arrival times of trucks problem is then addressed in Boywitz and Boysen (2018) for the front access deep-lane system.

2.4 Positioning our problem with respect to the literature

The problem presented in this paper shares some features with storage assignment problems from the literature. Nevertheless, they have never been considered jointly in a unique setting. Firstly, for each product type, storage location assignment decisions are taken simultaneously with sequencing decisions about the order of using the assigned storage locations. The aim is twofold: *i*) to enable a FIFO retrieval policy among storage locations later on, separately per product type, and *ii*) to exploit additional storage availability on the top of pairs of consecutive storage locations along each sequence. To the best of our knowledge, the combination of such features has never been addressed before in the storage assignment literature.

Moreover, the considered two-level storage policy is different than others from the literature. By considering in particular the triangular policy addressed in Zäpfel and Wasner (2006), reviewed in Section 2.2, there the only condition to locate a SKU at the upper level is that the lower locations are already occupied. Consequently, the number of positions potentially available at the upper level is fixed and only depends on the number of locations on the ground level. In our problem, instead, an additional condition has to be respected. Precisely, both the storage locations in each sequence (at the lower level) and the storage locations made available on the top of consecutive storage locations along the sequence must be assigned to the same product type. Therefore, the number of potentially available storage locations at the upper level is not fixed, rather it may change depending on the amount of SKUs to store and their product type. This also affects the calculation of the storage remaining available after the allocation of the SKUs.

3 The problem addressed

The problem is defined in a warehouse composed of a set \mathcal{D} of departments, each having a given number of storage locations. Let \mathcal{K} denote the set of the different product types requiring storage in a given time horizon (e.g., a day), and let q^k be the number of items of product type k that need storage, for each $k \in \mathcal{K}$. Each storage location i has a capacity which depends on the product type k , i.e., it allows the storage of at most c_i^k items, which must be all of the same type k .

The majority of the product types can be stored in any available storage location, i.e., a random storage policy is considered. However, often special product types do exist, which have to be preferably managed according to a dedicated storage policy. Precisely, we consider n subsets $\mathcal{K}_1^s, \dots, \mathcal{K}_n^s$ of special product types, that should be preferably stored in departments belonging to the subsets $\mathcal{D}_1^s, \dots, \mathcal{D}_n^s$, respectively.

As previously described, in addition to assign a set of storage locations to each product type $k \in \mathcal{K}$, a suitable sequencing of the assigned storage locations must be devised for each product type, i.e., it has to be decided the ordering with which the storage locations will be filled up during the storing operations to easily implement the FIFO picking criterion among storage locations and for each product type to follow in the successive order picking steps. Notice that a FIFO

picking criterion inside storage locations is not required instead, the picking criterion in that case depending on the characteristics of the used storage locations. Such sequencing decisions rely on a set \mathcal{P} , indicating what pairs of candidate locations can be consecutive in a sequence. \mathcal{P} is usually determined according to some Quality of Service considerations, for example the geographical position of the storage locations in the warehouse departments. Specifically, a storage location j can be the successor of a storage location i in a sequence if and only if $(i, j) \in \mathcal{P}$. Observe that the FIFO retrieval policy is typical of warehouses in application contexts such as the pharmaceutical and the tissue ones. The latter is the one addressed in the presented case study in Section 6.1.

In addition, for each product type, the selected sequencing also determines the availability of additional extra storage for that product type. Specifically, additional storage can be made available on the top of pairs of consecutive storage locations along the sequence. For each pair of consecutive storage locations along the sequence, say i and j , and for each product type k , the amount of the extra storage available on the top of i and j will be denoted by b_{ij}^k . That is, this amount does depend on the two storage locations at the ground level and on the product type to be stored. In particular, $b_{ij}^k = 0$ if i and j are not physically contiguous. Hereafter, this policy will be referred to as *sequence-based two-level storage policy*. Items stocked on the top of two storage locations will be picked up before the ones in the two storage locations at its bottom, so locally relaxing the FIFO picking criterion for stability motivations.

The problem consists in assigning to each product type $k \in \mathcal{K}$ a sequence of storage locations, among those available in the warehouse, in order to store all the q^k items needing storage, considering that items with product type in $\mathcal{K}_1^s, \dots, \mathcal{K}_n^s$, have preferable departments. In order to satisfy the storage demand of each product type $k \in \mathcal{K}$, the sum of the capacities of the storage locations assigned to k plus the extra storage made available for k on the top level must be greater than or equal to q^k .

By summarizing, the addressed problem consists in assigning a sequence of available storage locations to each product type, by satisfying the following constraints:

- each storage location can be assigned to a unique product type,
- the sum of the capacities of the storage locations assigned to a product type plus the extra storage made available for it on the top level must be greater than or equal to the storage demand of the product type,
- special product types should be preferably stored in the related, specific, departments,

while maximizing the residual storage capacity which remains available after the assignment of the storage locations.

Theorem 3.1 *The assignment and sequencing storage location problem under the sequence-based two-level storage policy is NP-Hard.*

Proof: The proof is by reduction from the Maximum Fixed-Length Disjoint Paths Problem (MFLDP), which is NP-complete in its decisional form (Garey and Johnson, 1979). Consider an instance of MFLDP, say P_1 . Given a directed graph $G = (V, E)$, vertices s and t , and positive integers $K, Q \leq |V|$, P_1 asks to verify whether G contains K mutually vertex-disjoint directed paths from s to t , each involving exactly Q arcs.

Now, given P_1 , defines the following instance of the assignment and sequencing storage location problem under the sequence-based two-level storage policy, hereafter denoted as P_2 . In P_2 , there is an available storage location for each node $i \in V$. We will refer to it as i , too. Moreover, K product types require storage in P_2 (i.e., $|\mathcal{K}| = K$), each with demand $Q - 1$ (i.e., $q^k = Q - 1, \forall k \in \mathcal{K}$). No special product types are given (i.e., $\mathcal{K}_1^s = \emptyset, \dots, \mathcal{K}_n^s = \emptyset$). The capacity of each storage location is 1 independently of the product type (i.e., $c_i^k = 1, \forall k \in \mathcal{K}$ and each storage location i). Finally, $\mathcal{P} = E$, and therefore a storage location j can be the successor of a storage location i in any sequence if and only if $(i, j) \in E$, with parameters $b_{ij}^k = 0, \forall k \in \mathcal{K}$ (meaning, for example, that no two pairs of available storage locations are contiguous in P_2).

By construction, G contains K mutually vertex-disjoint directed paths from s to t , each involving exactly Q arcs, i.e., the instance P_1 is feasible, if and only if in instance P_2 it is possible to assign a sequence of storage locations to each product type by satisfying its demand, i.e., $Q - 1$. Equivalently, if and only if the residual capacity of the warehouse after the assignment is $|V| - (Q - 1)K$. The thesis follows ¹ \square

Since the sequence-based two-level storage policy has no role in the proof of Theorem 3.1, we also get:

Corollary 3.1 *The relaxation of the assignment and sequencing storage location problem under the sequence-based two-level storage policy, where storing on the upper level is forbidden, is NP-Hard, too.*

4 A multicommodity flow model

In this section we propose a multicommodity flow formulation to the assignment and sequencing storage location problem described in Section 3. The used notation is summarized here for the sake of completeness. We define the set \mathcal{K} of the different product types to be stored in a given time horizon, and the n subsets of \mathcal{K} of special products, $\mathcal{K}_1^s, \dots, \mathcal{K}_n^s$. Moreover, we define the set \mathcal{D} of all the departments of the warehouse, considering that special products in $\mathcal{K}_1^s, \dots, \mathcal{K}_n^s$ should be preferably stored in departments belonging to the subsets $\mathcal{D}_1^s, \dots, \mathcal{D}_n^s$, respectively. Finally, q^k denotes the number of items of product type $k \in \mathcal{K}$ to store in the warehouse.

4.1 The auxiliary graph

In order to formulate the problem, we introduce an auxiliary graph $G = (N, A)$ describing the current availability of storage locations in the warehouse. This graph reformulation is in some way inspired by the proof of NP-Hardness presented in Section 3.

The set of nodes N consists of:

- a set \mathcal{S} containing one node for each storage location available in the warehouse at the beginning of the time horizon;
- a fictitious source node Σ ;

¹MFLDP aims at defining K or more disjointed paths. Nevertheless, also the variant of this problem where exactly K disjointed paths are sought is NP-complete.

- a fictitious sink node Θ .

A balance c_i^k is associated with each node $i \in \mathcal{S}$ for each $k \in \mathcal{K}$, which gives the availability (in terms of items) of the corresponding storage location for product type k . The set of arcs A is defined in order to model the assignment of a sequence of storage locations to each product type in \mathcal{K} . As indicated before, the modeling idea is to associate a suitable sequence of available storage locations with each $k \in \mathcal{K}$, thus specifying in which order the storage locations have to be used when storing and, consequently, the order with which storage locations will be emptied for shipping operations later on. Specifically, the set of arcs A consists of:

- arcs (Σ, j) , with $j \in \mathcal{S}$, to model the assignment of the first storage location to a product type;
- arcs (i, j) , with $i, j \in \mathcal{S}$, $(i, j) \in \mathcal{P}$, to model the assignment of the available storage location j immediately after the available storage location i ; recall that \mathcal{P} is the allowed set of consecutive storage locations along a sequence;
- arcs (i, Θ) , with $i \in \mathcal{S}$, to model the assignment of the last storage location to a product type.

In particular, the subset of arcs connecting two contiguous storage locations will be denoted by

$$A_c = \{(i, j) \in A : \text{storage locations } i \text{ and } j \text{ are contiguous}\}.$$

A weight b_{ij}^k is associated with each arc (i, j) in A for each product type k in K , which indicates the amount of the extra storage made available on top of i and j for product k . Notice that $b_{ij}^k = 0$ for each k if $(i, j) \notin A_c$.

Example 4.1 Consider a warehouse composed of three blocks of storage locations, as shown in Figure 1a. The two blocks on the left contain three storage locations each at the ground level, while the block on the right contains eight storage locations at the ground level. In each storage location four items may be stored, independently of the product type. Positions are depicted as rectangles within each storage location and are full black colored if occupied by items of some product types, or white if empty and available for storage. Suppose that in a given day, a certain amount of items need to be stored. In that case, the storage locations of interest for the addressed instance are four and they are outlined in the figure by an identifier inside a gray circle. Figure 1b reports

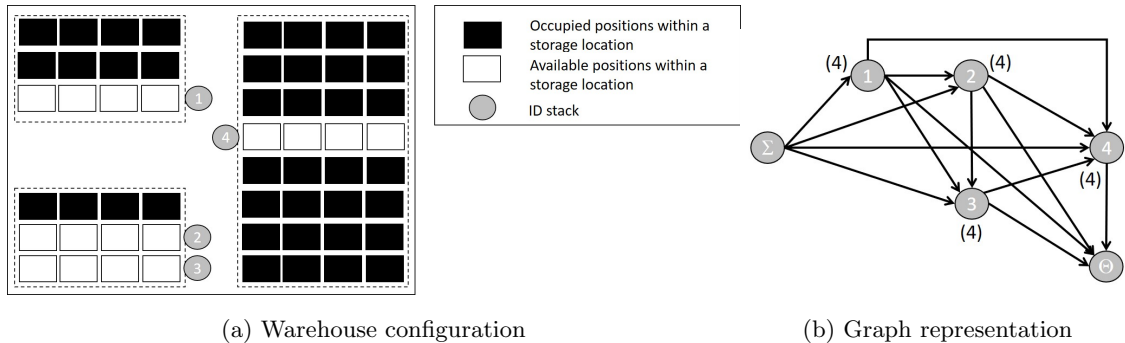


Figure 1: Warehouse configuration and related graph representation

the auxiliary graph associated with the considered instance. The nodes are marked with the same identifiers used in Figure 1a and the corresponding balances are associated between brackets.

4.2 Sequence of assigned storage locations as a directed path

We model the sequence of storage locations to assign to each product type $k \in \mathcal{K}$ by means of a directed path in the auxiliary graph from node Σ to node Θ , along which the quantity q^k to be stocked in the warehouse is sent, thus modeling the storing operations at the selected storage locations. This sending is formulated in terms of a flow pushed along the path. However, this is a non traditional flow pushing. In fact, the amount of flow of product type k entering a node i along the path is suitably decreased, according to the capacity of the involved storage location, in order to model a storing operation at that storage location. Specifically, the flow is decreased by c_i^k , unless the predecessor of i along the path represents a storage location which is contiguous to i . In the latter case, the flow of product type k is decreased by $c_i + b_{ij}^k$, because of the sequence-based two-level storage policy previously described. A progressive flow reduction so gives rise for each $k \in \mathcal{K}$.

Example 4.1 (continued). *Considering again the Example 4.1, suppose that a storage assignment needs to be solved to store 8 and 10 items of two different product types, i.e., product type 1 and product type 2, respectively. A possible solution is given in Figure 2. The two paths corresponding to the product type 1 and 2 are drawn with different dashed lines. The flow along each path is also reported, modeling the storing operations along the corresponding storage location. Considering the product type 1, a flow 8 is pushed along the corresponding path, since 8 items need to be stored. The first storing operation is performed at storage location 1. The latter is saturated, i.e., 4 items are stored, and the remaining 4 move along the path to be stocked in storage location 4, which is the second in the sequence. As for product type 2, a flow equal to 10 is pushed along the corresponding path, due to its storage request. The first assigned storage location is 2, where 4 items are stocked. The remaining 6 items enter the second assigned storage location, number 3, in terms of a flow equal to 6. Since storage locations 2 and 3 are physically contiguous (see Figure 1a), the additional upper capacity is exploited, and the 6 items are all stocked in storage location 3 (4 items) and on the top of storage locations 2 and 3 (2 items). Notice that a negative flow equal to*

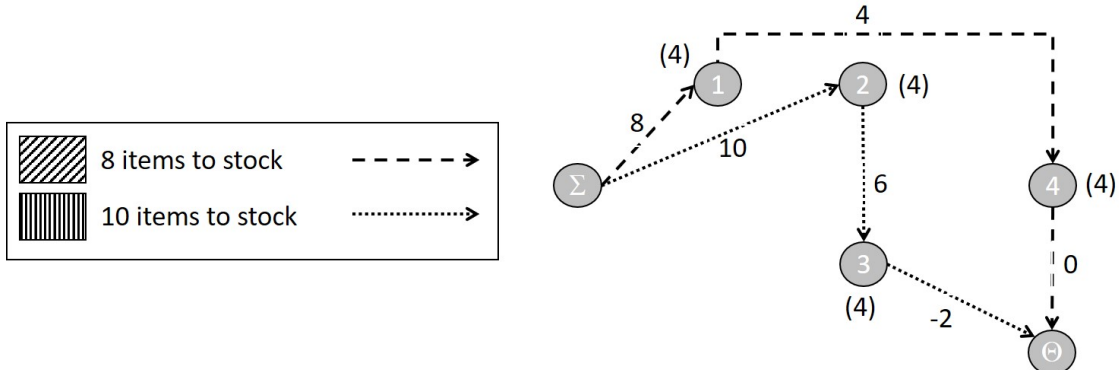


Figure 2: Solution representation in terms of directed paths.

-2 is associated with the last arc of the path, representing the opposite of the storage availability of the last storage location used.

4.3 The objective function

The goal of the addressed problem is to maximize the available storage capacity after the assignment of the storage locations, which may be a very critical objective in several application contexts, due to the huge number and volume of movements which are usually performed. Notice that maximizing the available storage capacity is not equivalent to minimize the number of assigned storage locations, when a longer time horizon is taken into account. This is due to the sequence-based two-level storage policy previously discussed.

Figure 3 highlights the difference between the two objectives. Consider two consecutive days

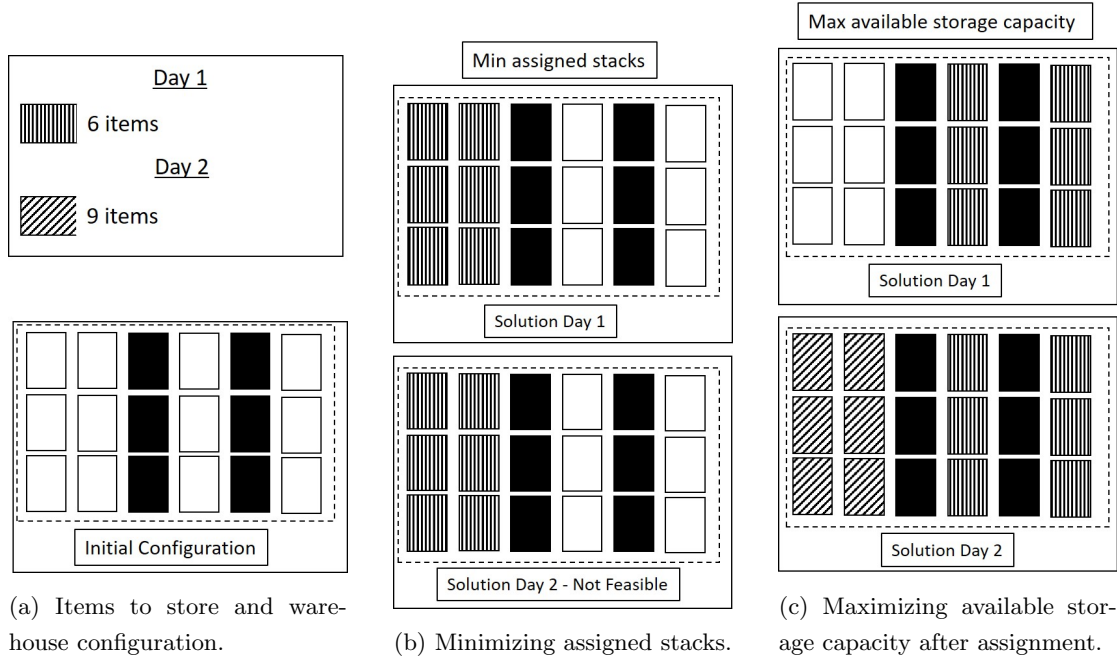


Figure 3: Difference between minimizing assigned stacks (b) and maximizing available storage capacity after assignment (c).

in which, respectively, 6 and 9 items of two different product types need to be stored, and the initial configuration of the warehouse reported in Figure 3a. Figure 3b shows an optimal solution in case the minimization of the assigned storage locations is the objective of the problem. The items needing storage in day 1 are stocked into storage location 1 and 2; as a result, there is not enough space available for storing items of day 2. Figure 3c describes instead an optimal solution in case the available room in the warehouse after the assignment is maximized. The items needing storage in day 1 are stocked into storage location 4 and 6. The available potential room after this assignment consists of 6 columns at the ground level plus 3 columns on top of these two. The items needing storage in day 2 can then be stocked into storage location 1 and 2 by exploiting the upper capacity made available by occupying two contiguous locations at the ground level with items of the same type. No unfeasibility is thus achieved in day 2.

4.4 Model description

To formulate the addressed problem, we introduce three families of variables:

- $x_{ij}^k \in \{0, 1\}$, for any $k \in \mathcal{K}$ and $(i, j) \in A$, model the directed path selected for product type k in terms of a unitary flow from node Σ to node Θ ;
- $f_{ij}^k \in \mathbb{R}$, for any $k \in \mathcal{K}$ and $(i, j) \in A$, model the storing operations along the sequence of storage locations assigned to k in terms of a suitable flow along the path;
- $y_{ij} \in \{0, 1\}$, for any $(i, j) \in A_c$, model the sequence-based two-level storage policy for the arcs connecting available contiguous storage locations.

Using these variables, the proposed multicommodity flow model can be stated as follows:

$$\max \sum_{i \in \mathcal{S}} c_i^{max} \left(1 - \sum_{k \in \mathcal{K}} \sum_{j \in N^+(i)} x_{ij}^k \right) + \sum_{(i,j) \in A_c} b_{ij}^{max} y_{ij} + \sum_{k \in \cup_{p=1, \dots, n} \mathcal{K}_p^s} \sum_{(i,j) \in A} \delta_{ij}^k x_{ij}^k \quad (1)$$

subject to:

$$\sum_{j \in N^-(i)} x_{ji}^k - \sum_{j \in N^+(i)} x_{ij}^k = \begin{cases} -1 & \text{if } i = \Sigma \\ 0 & \text{if } i \in \mathcal{S} \\ 1 & \text{if } i = \Theta \end{cases} \quad \forall k \in \mathcal{K}, \forall i \in N, \quad (2)$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in N^-(i)} x_{ji}^k \leq 1 \quad \forall i \in \mathcal{S}, \quad (3)$$

$$\sum_{j \in N^+(\Sigma)} f_{\Sigma j}^k = q^k \quad \forall k \in \mathcal{K}, \quad (4)$$

$$\sum_{j \in N^-(i)} f_{ji}^k - \sum_{j \in N^+(i)} f_{ij}^k = \sum_{j \in N^-(i)} c_i^k x_{ji}^k + \sum_{\substack{j \in N^-(i): \\ (j,i) \in A_c}} b_{ji}^k x_{ji}^k \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{S}, \quad (5)$$

$$0 \leq f_{ij}^k \leq q^k x_{ij}^k \quad \forall k \in \mathcal{K}, \forall (i, j) \in A : j \neq \Theta, \quad (6)$$

$$-2c_{max} x_{i\Theta}^k \leq f_{i\Theta}^k \leq 0 \quad \forall k \in \mathcal{K}, \forall (i, \Theta) \in A, \quad (7)$$

$$y_{ij} \leq 1 - \sum_{k \in \mathcal{K}} \sum_{l \in N^+(i)} x_{il}^k \quad \forall (i, j) \in A_c, \quad (8)$$

$$y_{ij} \leq 1 - \sum_{k \in \mathcal{K}} \sum_{l \in N^+(j)} x_{jl}^k \quad \forall (i, j) \in A_c. \quad (9)$$

The objective function (1) consists of two parts. The first two summations define the primary optimization goal, i.e., the available storage capacity after the assignment, to be maximized, while the last sum aims to satisfy a secondary optimization goal, i.e., the request that special products should be preferably stored in specific departments, which is thus pursued as a soft constraint. Specifically, the first sum equals to the capacity of the storage locations which have not been assigned, i.e., the corresponding nodes do not belong to any path. Here c_i^{max} does denote the maximum among the c_i^k capacities of storage location i . The second sum counts the extra storage which can be made available on the top level, thanks to the sequence-based two-level storage policy. Recall that A_c is the set of arcs connecting two contiguous storage locations, and each of them may contribute to the available storage capacity, via extra storage on their top level,

whenever both i and j are empty unassigned storage locations, as guaranteed by constraints (8)-(9) which are based on the auxiliary binary variables y_{ij} . Here b_{ij}^{max} does denote the maximum among the b_{ij}^k availabilities for the arc (i, j) . The first part of the objective function thus models the maximum possible residual capacity of the warehouse after the storing of the current product types, whose full utilization will however depend on the future product types to be stored and their related demands. Notice that only empty storage locations are considered in the residual capacity calculation, since they can be assigned to any product type. On the contrary, partially occupied storage locations are disregarded, since each of them is constrained to be assigned to the (unique) product type currently associated with it. The third sum involves parameters δ_{ij}^k which are set in such a way that it is more convenient to assign storage locations belonging to the department sets $\mathcal{D}_1^s, \dots, \mathcal{D}_n^s$ to the product types in $\mathcal{K}_1^s, \dots, \mathcal{K}_n^s$, respectively.

Constraints (2) define a directed path from Σ to Θ for each $k \in \mathcal{K}$ by means of the binary variables x_{ij}^k , with the aim of modeling the assignment of a sequence of storage locations to each k . In a standard way, each path is modeled in terms of a unitary flow sent from the source node Σ to the destination node Θ . $N^+(i)$ and $N^-(i)$ denote the sets of nodes linked to i via an exiting and an entering arc, respectively, that is

$$N^+(i) = \{j \in N : \exists (i, j) \in A\}, \quad N^-(i) = \{j \in N : \exists (j, i) \in A\}. \quad (10)$$

Constraints (3) state that each storage location can be assigned to at most one product type $k \in \mathcal{K}$. Constraints (4)-(5) are the flow conservation constraints, related to the flow variables f_{ij}^k , which are used to model the storing operations along the sequence of storage locations assigned to each $k \in \mathcal{K}$. Notice that (5) also model the sequence-based two-level storage policy via the second addendum of the right hand side. In fact, if j is contiguous to i along the path, i.e., $(j, i) \in A_c$, then an extra storage b_{ji}^k can be exploited on the upper level, above j and i , depending on k . Otherwise, the capacity to be considered for the storage location i is only c_i^k . Constraints (6)-(7) link together assignment and flow variables. Specifically, a flow of type k can be sent along an arc (i, j) only if (i, j) belongs to the path assigned to k . In that case, if $(i, j) \in A$ with $j \neq \Theta$, i.e., f_{ij}^k represents the amount of product k still to be stored after the storing operations at i , then such an amount must be nonnegative and less or equal to the total amount q^k to be stored. As for the arcs of type (i, Θ) , since they model the end of the storing operations, their flow must be nonpositive and bounded from below by $-c_{max}$, where c_{max} denotes the maximum capacity of all storage locations in the warehouse. Notice that, for each k , the opposite of the flow along (i, Θ) , i.e., $-f_{i\Theta}^k$, indicates the number of items of product k which can be still stored in i , thus providing an useful additional information about the status of the warehouse in terms of storage availability.

The sets, the parameters and the variables related to the model are summarized in Table 1.

4.5 Enhanced formulations

In order to enhance the sequence-based two-level storage policy, we have analyzed and experimented two alternative formulations, both based on model (1)-(9). The first one concerns the addition of a set of valid inequalities, hereafter referred to as *top level constraints*, to (2)-(9), while the second formulation has been obtained by adding a fourth term to the objective function (1).

The top level constraints state that, whenever two contiguous storage locations i and j are assigned to a certain product type k , then j must be the successor of i along the path associated

Table 1: Sets, parameters and variables used in the model.

Sets	
\mathcal{K}	set of different product types
$\mathcal{K}_1^s, \dots, \mathcal{K}_n^s$	subsets of special product types
$\mathcal{D}_1^s, \dots, \mathcal{D}_n^s$	subsets of departments where special product types have to be preferably stored
\mathcal{S}	set of available storage locations
A_c	set of arcs connecting pairs of contiguous storage locations
Parameters	
Σ	fictitious source node
Θ	fictitious sink node
c_i^k	capacity of storage location i for product type k
b_{ij}^k	capacity made available on top of i and j for product type k , if $(i, j) \in A_c$
δ_{ij}^k	storing requisite weights for i, j and product type $k \in \cup_{p=1, \dots, n} \mathcal{K}_p^s$
q^k	number of items to stock for product $k \in \mathcal{K}$
c_i^{max}	maximum capacity of storage location i
b_{ij}^{max}	maximum availability of arc (i, j)
c_{max}	maximum capacity of all storage locations in the warehouse
Variables	
$x_{ij}^k \in \{0, 1\}$	model the directed path from node Σ to node Θ for product type k
$f_{ij}^k \in \mathbb{R}$	model the storing operations along the sequence of storage locations assigned to k
$y_{ij} \in \{0, 1\}$	model the sequence-based two-level storage policy

with k . The rationale is to be able to exploit the additional extra storage that can be created on the top of i and j , with a benefit for the total amount of storage available after the assignment. Formally, the top level constraints are defined as follows:

$$x_{ij}^k \geq \sum_{h \in N^+(i)} x_{ih}^k + \sum_{h \in N^+(j)} x_{jh}^k - 1 \quad \forall k \in \mathcal{K}, \forall (i, j) \in A_c. \quad (11)$$

We will refer to the model (1)-(9) enhanced with constraints (11) as SLAP-C.

The second formulation differs from (1)-(9) only in the objective function. Precisely, the following fourth summation is added to (1):

$$\sum_{k \in \mathcal{K}} \sum_{(i, j) \in A_c} x_{ij}^k. \quad (12)$$

The rationale is analogous to the one expressed by constraints (11), i.e., to increase the number of pairs of storage locations which are both physically contiguous and consecutive along the paths, so as to exploit the two level storage. We will refer to the model (1)-(9) with the additional term (12) in the objective function as SLAP-O.

The two enhanced formulations have been numerically evaluated and compared, as shown in Section 6.

5 A matheuristic approach

Preliminary experiments using the state-of-the-art commercial solver CPLEX have shown that the running time to solve the enhanced formulations presented in Section 4.5 can be higher than the time limit usually required in the reference application context, which is about one hour. In particular, a huge amount of time was spent by the solver to find a first feasible solution to start the resolution process. In order to overcome this issue, we propose the following two-phase matheuristic approach:

Phase 1: generating an initial feasible solution to provide to the solver;

Phase 2: calling CPLEX to solve either SLAP-C or SLAP-O (see Section 4.5), starting from the computed initial solution.

Phase 1 consists of the following steps:

1. The list of the product types to store during the planning horizon is rearranged in a nonincreasing order with respect to the number of items to store.
2. The ordered list is partitioned into Λ subsets, where Λ is a parameter of the approach, in such a way that each group contains about the same number of items to be stored, i.e., approximately $(\sum_{k \in \mathcal{K}} q^k)/\Lambda$.
3. The set \mathcal{K} is partitioned into Λ subsets accordingly. Let \mathcal{K}_λ , with $\lambda = 1, \dots, \Lambda$, denote the resulting subsets.
4. Λ subproblems are generated and solved in cascade to stock the subsets of product types in $\mathcal{K}_1, \dots, \mathcal{K}_\Lambda$, respectively, each time removing those storage locations already assigned in the previous solved subproblems. Specifically, by denoting by Φ_λ the set of storage locations assigned to the product types in \mathcal{K}_λ when solving the λ -th subproblem, then $\Phi_1 \cup \dots \cup \Phi_{\lambda-1}$ are removed from \mathcal{S} when determining Φ_λ . So doing, it is not possible to assign the same storage location in different subproblems. Notice that, if the SLAP-C formulation is selected to solve the overall problem, then all the Λ subproblems which are generated to determine an initial feasible solution are solved via SLAP-C. Otherwise, i.e., SLAP-O is used, then the Λ subproblems are solved using SLAP-O.
5. The optimal solutions of the Λ subproblems are merged into a unique solution, which is the initial feasible solution provided to the Branch and Bound algorithm of CPLEX.

The matheuristic approach is summarized in Algorithm 1.

6 Numerical experiments

6.1 The case study addressed

The production site of the considered company, leader in the tissue sector, is composed of a production area, a warehouse, a sortation area and several shipping docks. The warehouse is larger than 10,000 m² and comprises four departments. Each department has a rectangular internal layout composed of blocks of storage locations, which are stacks in the considered application.

Algorithm 1 The matheuristic approach

- 1: Sort \mathcal{K} in a nonincreasing order with respect to the number of items to stock
 - 2: Partition the ordered set \mathcal{K} into Λ almost balanced groups: $\mathcal{K}_1, \dots, \mathcal{K}_\Lambda$
 - 3: $\Phi_0 = \emptyset$
 - 4: **for** $\lambda = 1, \dots, \Lambda$ **do**
 - 5: Remove $\Phi_{\lambda-1}$ from \mathcal{S}
 - 6: Solve the λ^{th} subproblem on K_λ
 - 7: Insert into Φ_λ the assigned storage locations
 - 8: **end for**
 - 9: Unify the subproblem solutions: $\Phi = \cup_\lambda \Phi_\lambda$
 - 10: Solve the original problem starting from solution Φ
-

Stacks are accessible only frontally, and they are framed by narrow storage aisles and wide cross aisles. Different blocks may be composed of different number of stacks, all having though the same capacity. However, stacks belonging to different blocks may have different capacities. Specifically, the storage area is divided into 29 blocks, which are composed of a variable number of stacks ranging from 15 to 65. Stacks have a capacity ranging from 8 to 17 items at the ground level, independently on the product type to store.

The sortation area is used as a collection area where items can be temporarily stocked, once retrieved from their positions within the warehouse, waiting to be loaded on the trucks. It can stock up to 1,000 items, and is normally filled up as much as possible during the night to quickly start the truck loading operations the next morning. The structure of the warehouse is depicted in Figure 4a, while the internal structure of a department is depicted in Figure 4b.

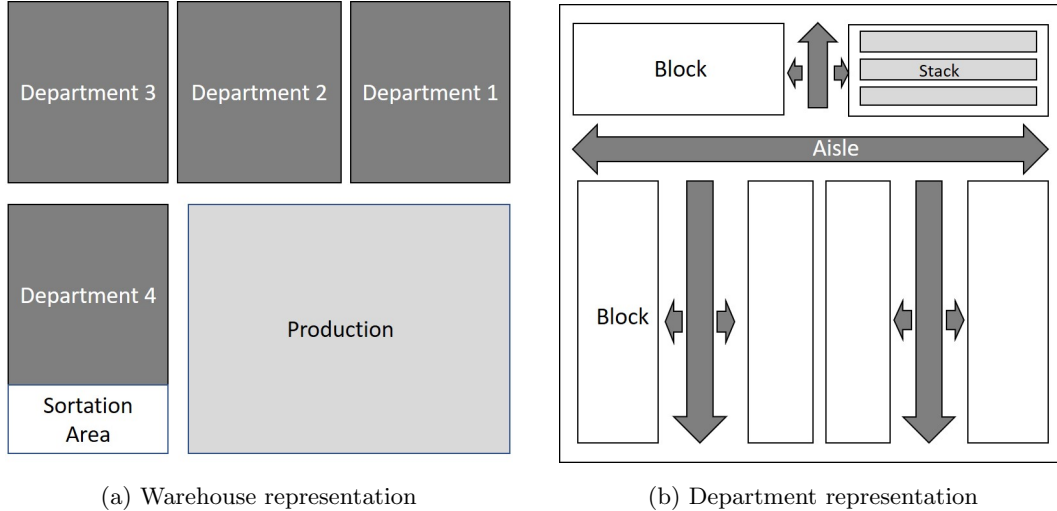


Figure 4: Warehouse and department representations

The production site works daily on 3 shifts of 8 hours. More than 300 different product types can be produced. Two sets of special product types are considered, i.e., high rotational and perishable products. High rotational and perishable products have a favorite department, as in the general presentation in Section 3. Precisely, and referring to Figure 4a, department 1 should

be preferable dedicated to perishable products, while department 4, which is a kind of fast picking area since it is located near the sortation area, should be preferable dedicated to high rotational products. Hereafter perishable products, i.e., tissues in the considered context, will be denoted as P, while high rotational products will be denoted as HR. Items are released by the production area in small quantities and constantly during each shift, already wrapped and arranged in so-called *columns* on pallets all having the same dimension. That is, items are columns in the addressed case study, and therefore the inventory will be expressed in terms of columns.

The planning horizon we consider in our application context is the day. On average, about 900 columns are released per day (300 on average for each shift) and about the same quantity is shipped. Departments and sortation area are thus steadily near to their saturation level. The list of the product types released per day, together with the associated number of columns to store, is known and will be referred to as the *storage list* in the following. As described in Section 3, for each product type included in the daily storage list, a sequence of empty stacks available in the warehouse must be assigned to the product type, which must be suitable to guarantee the storage of all the columns of that product type. Stack assignment decisions must respect the rules previously introduced.

Moreover, in the considered application context a partially occupied stack may exist for each product type. If this verifies, the partially occupied stack must be assigned to the product type, and it must be the first stack involved by storing operations for the considered product type. It is important to emphasize that, for each product type, at most one partially occupied stack may exist in the warehouse for storage assignment decisions. This is due to the fact that, as described in Section 3, stacks occupied by columns of the same product type are filled one at a time, according to the sequence established. Picking operations are then performed by picking columns from one stack at a time following this order, from the oldest stack to the newest one. This causes that at most one partially occupied stack per product type may exist each day. Consequently, denoting by \mathcal{K}_o the set of the product types for which a partially occupied stack exists, and denoting by s^k the partially occupied stack of $k \in \mathcal{K}_o$, the auxiliary graph has been extended by adding one node for each element in \mathcal{K}_o , and the mathematical formulation (1)-(9) has been extended by inserting the following additional family of constraints:

$$x_{\Sigma, s^k}^k = 1 \quad \forall k \in \mathcal{K}_o. \quad (13)$$

A further relevant characteristic, discussed in Section 3, is that whenever two contiguous stacks are occupied by columns of the same product type at the ground level, an additional stack can be created on the top of the two stacks, having the same capacity of each ground stack at the basis. The previously defined sequence-based two-level storage policy has thus to be taken into account. As presented in Section 3, the goal is to maximize the residual capacity of the warehouse.

6.2 Plan of the experiments

Two types of experiments have been performed. In Section 6.4, we analyze the performance of the matheuristic approach presented in Section 5 on a daily basis, using either SLAP-C or SLAP-O as the kernel formulation of the approach, and varying the setting of the matheuristic parameters. The efficiency and the efficacy of the approach have been tested on a wide pool of real instances

related to the addressed case study, with the aim of identifying suitable parameter settings. The instances are described in Section 6.3. Then, in Section 6.5 we assess the efficiency and the efficacy of the approach in a wider and more complete setting, where assignment and sequencing storage location problems are consecutively solved, on a daily basis, jointly with picking and housekeeping operations. Specifically, we have simulated the use of the matheuristic approach on a 6-day week, by considering the kernel formulation and the setting of the parameters suggested by the first type of experiments. The matheuristic approach has been implemented using the OPL language and solved via CPLEX 12.6 solver (IBM ILOG, 2016). All the experiments have been conducted on an Intel Xeon 5120 computer with 2.20 GHz and 32 GB of RAM.

6.3 The instances

The data set provided by the company comprises the following daily information: the warehouse configuration at the beginning of the considered day (i.e., product types and corresponding number of columns inside the warehouse), the storage list of the day (i.e., product types and corresponding number of columns needing storage that day) and the shipping list of the day (i.e., product types and number of columns leaving the warehouse that day).

For the first type of experiments, we randomly selected 20 not consecutive days from the data set, and partitioned them into two classes depending on the total amount of columns to stock. The first class, called ClassHA, contains 10 days where the number of columns to stock is higher than the average number of columns to stock over the 20 selected days. The second class, called ClassLA, contains 10 days where the number of columns to stock is lower than the average. More in detail, the instances in ClassHA have to assign 1150 columns of 14 different product types on average: 1.2% are columns of type P, whereas 21.2% are columns of type HR. The instances in ClassLA have to assign 787 columns of 11 different product types on average: 1.5% are columns of type P, whereas 40.7% are columns of type HR. The average number of empty stacks before the assignment is similar in the two classes: 139 in ClassHA and 147 in ClassLA. Therefore, the size of the auxiliary graph used to model the problem in terms of multicommodity flows is about the same on average for the two classes of instances. However, the instances in ClassHA have more columns to assign, which are related to a greater number of different product types. This implies more commodities to manage on average in the corresponding MILP formulation, and so more paths to design. These characteristics render the stack assignment and sequencing particularly heavy for ClassHA, as reported next.

As an additional information, the average number of variables over the 20 instances is 292,462, while the average number of constraints is 441,715 for formulation SLAP-O and 443,411 for formulation SLAP-C.

6.4 Efficacy and efficiency of the matheuristic approach

The matheuristic approach relies either on the SLAP-O or on the SLAP-C formulation, and it is characterized by some parameters.

As described in Section 5, the matheuristic consists of two phases: Phase 1, devoted to the construction of an initial feasible solution through the resolution of Λ subproblems, and Phase 2, where the assignment and sequencing storage location problem is solved starting from the feasible solution determined in Phase 1. After some preliminary tests, we set $\Lambda = 5$.

Table 2: Performance of SLAP-O and SLAP-C formulations.

Formulation	Time limit	Optimality	Available space	% Columns	Runs
	Phase 1	gap	(columns)	in target Dep.	solved
SLAP-O	20	5.69%	2452	77.77%	60
SLAP-C	20	5.57%	2420	76.45%	58
SLAP-O	40	5.35%	2458	78.33%	60
SLAP-C	40	5.48%	2447	77.04%	60

Moreover, key parameters are the weights δ_{ij}^k . As outlined, in the experimented data set columns of type P should be preferably stored in department 1, whereas columns of type HR should be preferably stored in department 4. Increasing values of these parameters would tend to enhance the assignment of these special products to their target departments. After some preliminary tests, we decided to analyse the following three different settings for δ_{ij}^k :

$$\delta_{ij}^k = \begin{cases} 10, 25, 50 & \text{if } k \text{ is of type } P \text{ and } (i, j) \text{ links nodes in department 1,} \\ & \text{or } k \text{ is of type } HR \text{ and } (i, j) \text{ links nodes in department 4} \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we experimented two alternative ways to divide the time limit of 60 minutes, established by the warehouse managers to solve the problem, into the two phases of the matheuristic approach. We considered a first setting where 20 minutes are given to Phase 1 and 40 minutes to Phase 2, and a second setting where 40 minutes are given to Phase 1 and 20 minutes to Phase 2. In both settings, we split the time allocated to Phase 1 among the Λ subproblems proportionally with respect to the number of columns each subproblem has to handle. If a subproblem is solved before reaching the assigned time limit, then the remaining amount of time is added to the one allocated to Phase 2. The algorithm may stop the resolution of a subproblem before reaching the assigned time limit if the estimated percentage gap between the optimum and the current solution value is lower than 0.01%.

For both the versions of the matheuristic approach, based on SLAP-O or SLAP-C, respectively, the two time settings related to the two phases of the approach and the three settings for the weight parameters have been combined. Each of the 20 instances composing the data set has thus been solved 6 times by considering the SLAP-O formulation (120 runs) and 6 times by considering the SLAP-C formulation (120 runs), for a total of 240 runs.

We firstly investigated the impact of the used formulation (SLAP-O or SLAP-C) on the efficiency and efficacy of the matheuristic approach. Table 2 reports the average percentage optimality gap when the total time limit is reached, the average space available after the assignment of the products in the daily storage list (expressed in columns), the average percentage of columns of types P and HR assigned to their target departments, and the number of runs for which a solution was found within the time limit. The space available after the assignment has been calculated by summing up the capacities of the empty stacks at the ground level of the four departments plus, whenever two empty stacks are contiguous, the capacity of the stack that can be created on top of them.

The figures are reported separately for the two tested time settings, i.e., by distinguishing the runs where 20 and 40 minutes are allocated to Phase 1, respectively. Results show that SLAP-O strictly dominates SLAP-C regarding the average available space and the average number of columns of special product types in target departments, although the improvements are small on average. Moreover, the version of the matheuristic based on SLAP-C failed in finding a solution in two runs, when 20 minutes were allocated to Phase 1. According to these aggregated results, the version of the matheuristic based on SLAP-O seems to be more efficient and effective. Therefore, hereafter only results concerning the SLAP-O formulation will be reported.

Tables 3 and 4, for the instances in ClassLA and ClassHA, respectively, show the performance of SLAP-O in terms of average percentage optimality gap and average running time (in seconds), for the different combinations of the time setting for the two phases of the approach and of the weight parameters δ_{ij}^k .

Table 3: Performance of SLAP-O for ClassLA: average gaps and times.

δ_{ij}^k	Time limit Phase 1	Subproblem time (sec.)	% Subproblem gap	Overall time (sec.)	% Initial overall gap	% Final overall gap
10	20	0.01	7.87%	3339	33.81%	2.79%
10	40	0.01	7.87%	3330	33.81%	2.79%
25	20	13.11	7.84%	3535	10.21%	3.42%
25	40	22.53	7.82%	3488	10.18%	3.29%
50	20	18.50	8.07%	3508	15.70%	5.14%
50	40	31.39	7.95%	3443	15.67%	5.81%

Table 4: Performance of SLAP-O for ClassHA: average gaps and times.

δ_{ij}^k	Time limit Phase 1	Subproblem time (sec.)	% Subproblem gap	Overall time (sec.)	% Initial overall gap	% Final overall gap
10	20	22.97	6.39%	3485	30.41%	5.91%
10	40	31.67	6.18%	3442	30.38%	5.38%
25	20	22.93	7.16%	3486	11.56%	7.50%
25	40	19.35	6.17%	3503	9.74%	6.03%
50	20	18.12	6.17%	3510	13.67%	9.41%
50	40	32.22	6.26%	3439	13.60%	8.82%

Specifically, these figures are reported both for the subproblems solved to determine the first feasible solution in Phase 1, and for the overall approach. Regarding the overall approach, both the gap at the beginning of Phase 2 (% Initial Overall gap) and the one at the end of Phase 2 (% Final Overall gap) are reported.

The tables show that increasing the value of δ_{ij}^k generally makes the problem harder to solve on average. This is reasonable since the solver has to maximize the total space available after the assignment while trying to enhance the assignment of columns of types P and HR to their

Table 5: Performance of SLAP-O for ClassLA: available space and target departments.

δ_{ij}^k	Time limit Phase 1	Available space (columns)	% P in dep.1	% HR in dep.4
10	20	2897	83.89%	70.60%
10	40	2897	83.89%	70.60%
25	20	2879	89.93%	73.37%
25	40	2881	89.93%	73.37%
50	20	2824	91.95%	76.13%
50	40	2827	91.95%	76.13%

target departments, an objective which may be in contrast with the first one. Finally, observe that Phase 1 rapidly determines an initial feasible solution which, however, may be far from the optimum (see column % Initial Overall gap). The Phase 2 of the proposed approach, however, is able to strongly improve such an initial solution, as column % Final Overall gap testifies. This is especially true for the instances in ClassLA.

Moreover, for the same setting of δ_{ij}^k , allocating 40 minutes to Phase 1 and 20 minutes to Phase 2 makes the subproblem resolution more difficult, but usually has a positive impact on the solution of the overall approach in terms of percentage optimality gap.

Overall, the instances in ClassHA appear to be more difficult to address computationally, according to their characteristics outlined in Section 6.3.

Regarding the quality of the solutions returned by the matheuristic, Tables 5 and 6, for the instances in ClassLA and ClassHA, respectively, report the average space available in the warehouse after the assignment, the percentage of columns of type P assigned to department 1, and the percentage of columns of type HR assigned to department 4.

Table 5 shows that, for the instances in ClassLA, by increasing δ_{ij}^k the percentage of columns assigned to their target departments increases, at the expenses of the space available after the assignment, by confirming that maximizing the available space after the assignment and allocating special product types to target departments are more and more conflicting when δ_{ij}^k increase. In all the cases, however, the percentage of columns of types P and HR assigned to their target departments is very high, especially for P. Also notice that, on this class of instances, allocating 40 minutes to Phase 1 seems not to influence the special product types assignment, and shows just a marginal improvement regarding the available space after the assignment.

A similar trend can be observed for the hardest group of instances, i.e., the ones in ClassHA. Table 6 shows that, by increasing δ_{ij}^k , there is a decrease of the space available after the assignment versus an increase of the percentage of columns assigned to their target departments, which is very high for both product types P and HR. However, a less stable behavior can be observed for the latter figure, probably due to the hardness of the instances, which are more tricky to solve within the imposed time limit (see Table 4). Anyway, the percentages of columns of type HR stocked in department 4 are higher than for ClassLA, also because less columns of type HR has to be stocked on average in this class of instances. Finally, allocating 40 minutes to Phase 1 seems to be preferable in most cases.

Table 6: Performance of SLAP-O for ClassHA: available space and target departments.

δ_{ij}^k	Time limit Phase 1	Available space (columns)	% P in dep.1	% HR in dep.4
10	20	2060	85.89%	81.66%
10	40	2073	85.89%	82.15%
25	20	2047	90.80%	81.12%
25	40	2056	93.87%	81.95%
50	20	2006	92.64%	80.01%
50	40	2014	90.80%	82.19%

By summarizing, the reported results suggest the following observations:

- the SLAP-O formulation appears to be preferable to SLAP-C;
- increasing the value of δ_{ij}^k , SLAP-O seems to be computationally more difficult to solve on both ClassLA and ClassHA; in any case, ClassHA appears to be harder to address than ClassLA;
- increasing the value of δ_{ij}^k , the percentage of columns of types P and HR assigned to their target departments increases, at the expenses of the space available after the assignment; such a percentage is however very good also setting $\delta_{ij}^k = 10$ for k of product type P and for k of product type HR;
- allocating 40 minutes to Phase 1 seems to be preferable, especially for the instances in ClassHA.

6.5 Simulating stack assignment and sequencing over a week

In order to assess the impact of the proposed approach in a real logistic environment, we have analyzed its behavior by simulating the typical warehouse operations in a weekly time horizon. In the reference warehouse, described in Section 6.1, the following basic operations are performed on a daily basis:

- (i) the columns specified in the shipping list are picked from stacks according to the FIFO retrieving order, and are moved to the sortation area until its saturation; this is true except for columns in department 4, which are loaded directly on the trucks (see Figure 4a);
- (ii) a sequence of stacks is assigned to each product type in the storage list (the proposed matheuristic approach comes into play here), and the corresponding storage operations are performed accordingly;
- (iii) columns requested to leave the warehouse and not moved towards the sortation area before the assignment are now picked and loaded on trucks as well as the columns in department 4;
- (iv) some simple housekeeping operations are performed to enhance the space available in the warehouse, due to the high stock rotation index characterizing it.

We simulated the logistic process above as shown in Algorithm 2. We start with a realistic

Algorithm 2 One week simulation

Input data:

- Initial configuration for day 1
- Shipping lists for days 1 to 6
- Storage lists for days 1 to 6

- 1: **for** day $i = 1, \dots, 6$ **do**
 - 2: Fill the sortation area till saturation with products in the shipping list of day i
 - 3: Define the sets \mathcal{S} and \mathcal{S}_o accordingly
 - 4: Solve the assignment and sequencing problem for day i with Algorithm 1
 - 5: Fill the warehouse according to the algorithm solution
 - 6: Pick columns in the shipping list of day i , not already moved to the sortation area, according to the algorithm solution
 - 7: Pick columns in the shipping list of day i , which are in department 4, according to the algorithm solution
 - 8: Perform some housekeeping operations
 - 9: **end for**
-

configuration of the warehouse at day 1 (i.e., the first day of the considered week). The initial positions of the columns in the warehouse are randomly generated by respecting some agreed industrial practice or insights given by the company, in such a way as to start with a realistic configuration. At the beginning of each day i , columns of product types specified in the shipping list are picked one stack at a time, according to the FIFO retrieving order, and moved to the sortation area until its saturation (see line 2). As indicated before, the sortation area is normally filled up as much as possible during the night to quickly start the truck loading operations the next morning. Columns located in department 4 are not considered in this stage. The set \mathcal{S} of the empty stacks and the set, say \mathcal{S}_o , of the partially occupied stacks are thus defined (line 3). We solve the assignment and sequencing storage location problem related to day i by means of the proposed matheuristic approach (line 4). Then, the warehouse is updated by filling the assigned stacks in accordance with the solution found by the matheuristic approach (line 5), by picking those columns in the shipping list which could not be moved before to the sortation area due to its saturation (line 6), and by picking those columns in department 4 that need to be shipped (line 7). At the end of the day, some housekeeping operations are performed (line 8). Specifically, in order to exploit the sequence-based two-level storage policy, empty stacks are made contiguous within each block and, when possible, groups of occupied stacks are moved to other blocks of the warehouse. In addition, if a stack has less than the 80% of its capacity occupied at the ground level, then it is emptied and its columns are moved to a dummy storage area, which is present in the considered warehouse and is used just to perform such housekeeping movements. The final configuration obtained for day i is the initial configuration for the next day $i + 1$.

According to the results reported in Section 6.4, we used the matheuristic approach based on the SLAP-O formulation, by allocating 40 minutes to Phase 1, and we set $\delta_{ij}^k = 10$ for k in HR and (i, j) linking nodes in department 4, and for k in P and (i, j) linking nodes in department 1. The main motivation is that, working on a weekly basis and focusing on a large warehouse with a

Table 7: Storing and shipping lists for the selected week.

Day	Storage list			Shipping list		
	Total (col.)	P (col.)	HR (col.)	Total (col.)	P (col.)	HR (col.)
1	856	0	328	1078	8	541
2	1120	4	359	1282	18	556
3	1069	25	533	1212	22	624
4	1012	59	737	1094	21	497
5	811	6	670	947	5	521
6	156	0	103	0	0	0

very high rotation index, enhancing the available space after each daily stack assignment appears to be particularly crucial.

We report the results of the simulation process on a real data set of a week composed of three days in ClassLA and three days in ClassHA (shipping and storage operations are not planned on Sunday). Aggregated information on the storing and shipping lists for the considered week are shown in Table 7. In this week, storing and picking operations are quite balanced, although the total number of columns sent each day is usually greater than the total number of columns to stock. Regarding product of types P and HR, in some days more columns enter the warehouse compared to the ones leaving it, whereas in other days the opposite verifies.

Table 8 shows the results of the simulation for the selected week.

Table 8: Simulation on a week: number of columns in the warehouse and available space

Day	Before assignment		After assignment		After shipping		After housekeeping	
	Occupied	Available	Occupied	Available	Occupied	Available	Occupied	Available
1	16209	2789	17065	1861	15987	2189	15987	2270
2	15987	2809	17107	1501	15825	2049	15825	2143
3	15825	2742	16894	1527	15682	1908	15682	2062
4	15682	2759	16694	1502	15600	1851	15600	1956
5	15600	2106	16411	1190	15464	1498	15464	1559
6	15464	1559	15620	1359	15620	1359	15620	1361

For each day, the following indicators (in columns) are reported: the number of occupied and available columns immediately before the storage location assignment (i.e., after the execution of steps 2 and 3 in Algorithm 2); the number of occupied and available columns immediately after the storage location assignment (i.e., after the execution of steps 4 and 5 in Algorithm 2); the number of occupied and available columns after the shipping operations (i.e., after the execution of steps 6 and 7 in Algorithm 2); and the number of occupied and available columns after the housekeeping operations (i.e., after the execution of step 8 in Algorithm 2). The number of the available columns, i.e., the indicator Available, has been computed according to the specification of the objective function of the proposed mathematical formulation, i.e., (1). Precisely, we sum

the capacities of the storage locations which are not assigned to any product type, plus the extra storage which can be made available on the top level, thanks to the sequence-based two-level storage policy, for any pair of contiguous and unassigned storage locations. As in (1), only empty storage locations are considered in the availability calculation, since they can be assigned to any product type. On the contrary, partially occupied storage locations are disregarded, since each of them is constrained to be assigned to the (unique) product type currently associated with it.

The results in Table 8 show that the matheuristic approach seems to be successful in assigning and sequencing stacks also on a weekly basis. Notice, in particular, that since at most 21,299 columns can be stocked in the warehouse, in case a unique product type would be present, the considered week refers to a scenario where departments and sortation area are daily near to their saturation level. Assignment and sequencing operations may be therefore particularly difficult to address, so testifying the efficacy of the proposed matheuristic in computing solutions of high quality regarding the space availability. Also notice that the housekeeping operations seem to have just a marginal influence on the column availability.

Considering the secondary optimization goal of our approach, i.e., the assignment of columns of special products to specific departments, Table 9 reports the daily percentages of columns of product type P assigned to department 1 and of product type HR assigned to department 4.

Table 9: Percentage of columns of types P and HR in target departments in the daily solutions

day	1	2	3	4	5	6
P in dep.1	-	100%	64%	98%	83%	-
HR in dep.4	94%	89%	100%	36%	74%	98%

Interestingly, a high percentage of columns of type P is almost always assigned daily to department 1 (note that, there are no columns of type P to store in day 1 and 6), and a very high percentage of columns of type HR is assigned daily to department 4, as desired. This does not verify in day 4 for columns of type HR. This is due to the fact that department 4, when stack assignment is performed, is already almost full and only few stacks are available for storage, which are however selected in the associated solution.

The proposed matheuristic appears thus to be a valuable tool for solving the considered problem in a real application context.

7 Conclusions

In this paper, we address a problem which combines storage location assignment with sequencing decisions about the assigned storage locations, and which originates from a real-world application context in tissue logistics. Given a set of different product types, with the corresponding storage demand, a set of capacitated storage locations has to be assigned to each product type for the corresponding storing operations. In addition, a suitable sequencing of the assigned storage locations must be devised for each product type, i.e., it has to be decided the ordering with which the storage locations will be filled up during the storing operations. A motivation is that a FIFO picking criterion among storage locations is required per product type. The sequencing estab-

lished for the assigned storage locations will therefore allow to easily implement the FIFO policy in the successive order picking steps. Moreover, the selected sequencing may also determine the availability of additional extra storage for each product type, on the top of pairs of consecutive storage locations along the sequence. The goal is to maximize the storage capacity which remains available after the assignment of the storage locations. After proving its NP-Hardness, we model the problem as a constrained multicommodity flow problem on an auxiliary graph, and we propose a Mixed-Integer Linear Programming formulation, with some modeling enhancements, as well as a matheuristic approach based on the sequential resolution of multicommodity flow subproblems. A case study is then presented, which is related to the tissue logistics sector and which motivated our research in this topic. Computational experiments on a wide real test bed show the efficiency and the efficacy of the proposed approach.

We plan to extend the achieved results by studying an optimization problem which integrates the above described assignment and sequencing storage location decisions with a so called pick up and put away problem where, given a fleet of vehicles, decisions related to how and when to move SKUs in their assigned storage locations (put away) and to collect SKUs for shipping (pick up) need to be made. The assignment and sequencing of storage locations, the scheduling of put away and pick up operations, and the routing of the vehicles inside the warehouse define hard interdependent decisions which are very challenging to address.

Acknowledgements

The research has been supported by Region of Tuscany-Regional Government (POR FESR 2014-2020-Line 1-Research and Development Strategic Projects) through the Project IREAD4.0 under Grant CUP 7165.24052017.112000028.

References

- Accorsi, R., Baruffaldi, G., and Manzini, R. (2017). Design and manage deep lane storage system layout. an iterative decision-support model. *The International Journal of Advanced Manufacturing Technology*, 92(1-4):57–67.
- Ashayeri, J. and Gelders, L. (1985). Warehouse design optimization. *European Journal of Operational Research*, 21(3):285–294.
- Battini, D., Glock, C., Grosse, E., Persona, A., and Sgarbossa, F. (2016). Human energy expenditure in order picking storage assignment: A bi-objective method. *Computers & Industrial Engineering*, 94:147–157.
- Bortolini, M., Botti, L., Cascini, A., Gamberi, M., Mora, C., and Pilati, F. (2015). Unit-load storage assignment strategy for warehouses in seismic areas. *Computers & Industrial Engineering*, 87:481–490.
- Boysen, N., Boywitz, D., and Weidinger, F. (2018). Deep-lane storage of time-critical items: one-sided versus two-sided access. *OR Spectrum*, 40(4):1141–1170.

- Boywitz, D. and Boysen, N. (2018). Robust storage assignment in stack-and queue-based storage systems. *Computers & Operations Research*, 100:189–200.
- Carlo, H. J., Vis, I. F., and Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research*, 235(2):412–430.
- De Koster, R., Le-Duc, T., and Roodbergen, K. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.
- Dekker, R., Voogd, P., and Van Asperen, E. (2007). Advanced methods for container stacking. In *Container terminals and cargo systems*, pages 131–154. Springer.
- Ene, S., Küçükoğlu, İ., Aksoy, A., and Öztürk, N. (2016). A genetic algorithm for minimizing energy consumption in warehouses. *Energy*, 114:973–980.
- Foroughi, A., Boysen, N., Emde, S., and Schneider, M. (2020). High-density storage with mobile racks: Picker routing and product location. *Journal of the Operational Research Society*, pages 1–19.
- Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, CA.
- Gu, J., Goetschalckx, M., and McGinnis, L. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177(1):1–21.
- Gu, J., Goetschalckx, M., and McGinnis, L. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539–549.
- Hausman, W. H., Schwarz, L. B., and Graves, S. C. (1976). Optimal storage assignment in automatic warehousing systems. *Management Science*, 22(6):629–638.
- Larco, J., De Koster, R., Roodbergen, K., and Dul, J. (2017). Managing warehouse efficiency and worker discomfort through enhanced storage assignment decisions. *International Journal of Production Research*, 55(21):6407–6422.
- Lehnfeld, J. and Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239(2):297–312.
- Li, J., Moghaddam, M., and Nof, S. (2016). Dynamic storage assignment with product affinity and abc classification—a case study. *The International Journal of Advanced Manufacturing Technology*, 84(9-12):2179–2194.
- Maschietto, G. N., Ouazene, Y., Ravetti, M. G., de Souza, M. C., and Yalaoui, F. (2017). Crane scheduling problem with non-interference constraints in a steel coil distribution centre. *International Journal of Production Research*, 55(6):1607–1622.
- Meneghetti, A. and Monti, L. (2014). Multiple-weight unit load storage assignment strategies for energy-efficient automated warehouses. *International Journal of Logistics Research and Applications*, 17(4):304–322.

- Pang, K.-W. and Chan, H.-L. (2017). Data mining-based algorithm for storage location assignment in a randomised warehouse. *International Journal of Production Research*, 55(14):4035–4052.
- Quintanilla, S., Pérez, Á., Ballestín, F., and Lino, P. (2015). Heuristic algorithms for a storage location assignment problem in a chaotic warehouse. *Engineering Optimization*, 47(10):1405–1422.
- Ramtin, F. and Pazour, J. (2015). Product allocation problem for an AS/RS with multiple in-the-aisle pick positions. *IIE Transactions*, 47(12):1379–1396.
- Revillot-Narváez, D., Pérez-Galarce, F., and Álvarez-Miranda, E. (2020). Optimising the storage assignment and order-picking for the compact drive-in storage system. *International Journal of Production Research*, 58(22):6949–6969.
- Roodbergen, K. and Vis, I. (2009). A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194(2):343–362.
- Rouwenhorst, B., Reuter, B., Stockrahm, V., van Houtum, G.-J., Mantel, R., and Zijm, W. (2000). Warehouse design and control: Framework and literature review. *European Journal of Operational Research*, 122(3):515–533.
- Stahlbock, R. and Voß, S. (2008). Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52.
- Tang, L., Liu, J., Yang, F., Li, F., and Li, K. (2015). Modeling and solution for the ship stowage planning problem of coils in the steel industry. *Naval Research Logistics*, 62(7):564–581.
- Tang, L., Xie, X., and Liu, J. (2014). Crane scheduling in a warehouse storing steel coils. *IIE Transactions*, 46(3):267–282.
- Tang, L., Zhao, R., and Liu, J. (2012). Models and algorithms for shuffling problems in steel plants. *Naval Research Logistics*, 59(7):502–524.
- Vis, I. F. and De Koster, R. (2003). Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, 147(1):1–16.
- Xie, X., Zheng, Y., and Li, Y. (2014). Multi-crane scheduling in steel coil warehouse. *Expert Systems with Applications*, 41(6):2874–2885.
- Yang, P., Miao, L., Xue, Z., and Qin, L. (2015). An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems. *Journal of Intelligent Manufacturing*, 26(6):1145–1159.
- Zaerpour, N., Yu, Y., and de Koster, R. B. (2015). Storing fresh produce for fast retrieval in an automated compact cross-dock system. *Production and Operations Management*, 24(8):1266–1284.
- Zäpfel, G. and Wasner, M. (2006). Warehouse sequencing in the steel supply chain as a generalized job shop model. *International Journal of Production Economics*, 104(2):482–501.

Zhang, G., Nishi, T., Turner, S., Oga, K., and Li, X. (2017). An integrated strategy for a production planning and warehouse layout problem: Modeling and solution approaches. *Omega*, 68:85–94.